Prepared by Mohammad Umer

Python Tutorial

6.1 - Jump Statements in Python

Jump statements in Python are used to alter the flow of control in loops or conditional blocks.

There are three primary jump statements in Python:

- break
- continue
- return

- break statement

The break statement can be used within the loops. When the break statement is encountered, then the control is exited from the loop immediately i.e. it is `used for unconditional exit from the loop` and moves the control to the next statement after the loop.

Syntax: **break**

- Example

In [5]:
```
for i in range(1, 10):
    if i == 5:
        break      # Exit the loop when i is 5
    print(i, end=' ')
```

```
1 2 3 4
```

- continue Statement

The continue statement can be used within the loops. When the continue statement is encountered then the control bypassed the statements inside the loop & automatically passes to the beginning of the loop i.e. it is used to skip the particular iteration in a loop and continue to the next iteration.

Syntax: **continue**

- Example

In [4]:
```
for i in range(1, 10):
    if i == 5:
        continue        # Skip when i is 5
    print(i, end=' ')
```

```
1 2 3 4 6 7 8 9
```

- return Statement

The return statement is used to exit a function and optionally return a value to the caller.

Syntax: **return [expression]**

- Example

In [7]:
```
def add(a, b):
    return a + b  # Return the sum of a and b

result = add(2, 4)
result = add(2, 4)
print(result)
```

```
6
```

Q31. Write a Python program to check if a number is prime.

In [2]:
```
n = int(input("Enter Number : "))
count = 0

for i in range(1, n+1):
    if n%i==0:
        count += 1

if count==2:
    print(f"{n} is a Prime Number")

else:
    print(f"{n} is not a Prime Number")
```

```
22 is not a Prime Number
```

- OR -

In [7]:
```
n = int(input("Enter Number : "))
flag = 0

for i in range(2, n):                       # Looping Statement
    if n%i==0:
        flag = 1
        break                               # Jump Statement

if flag==0:                                 # Conditional Statement
    print(f"{n} is a Prime Number")

else:
    print(f"{n} is not a Prime Number")
```

```
12 is not a Prime Number
```

Both codes have same Time and space complexity. However, the second algorithm is slightly better in practice due to the early termination (break statement).

`Time Complexity: O(n)` and `Space Complexity: O(1)`

Q32. Write a Python program to display all prime numbers from 1 to N.

In [ ]:
```
n = int(input("Enter Number : "))

for i in range(2, n+1):             # as 1 is not a Prime number
    is_Prime = True                 # Boolean
    for j  in range(2, i):
        if i%j==0:
            is_Prime = False
            break

    if is_Prime:
        print(i, end=' ')
```

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

Q33. Write a Python program to display the number is a Power of 2 .

In [2]:
```
n = int(input("Enter number : "))
flag = 0
temp = n

while temp>1:
    if temp%2==1:
        flag = 1
        break

    else:
        temp = temp // 2

if flag==0:
    print(f"{n} is a power of 2")

else:
    print(f"{n} is not a power of 2")
```

```
8 is a power of 2
```